Seat No.: _____ *Enrollment No._____*

GUJARAT TECHNOLOGICAL UNIVERSITY

BE SEM-VIII Examination May 2012

Subject code: 181604

Subject Name: Design and Analysis of Algorithm (Department Elective-II)

Date: 08/05/2012 **Time: 10.30 am – 01.00 pm**

Total Marks: 70

Instructions:

1. **Attempt all questions.**
2. **Make suitable assumptions wherever necessary.**
3. **Figures to the right indicate full marks.**

Q.1 **(a)** Define an algorithm. What features does one look for in a good computer algorithm? When an algorithm is said to be correct? When an algorithm is said to efficient? Can an abstract algorithm be directly implemented? If yes, how? If not, what is its use? **06**

(b) How to compare the two algorithms? What do you mean by worst-case, average-case and best-case analysis of an algorithm? Is a Recursive algorithm less efficient compared to an Iterative one? Why or Why not? **04**

(c) Give an example where the choice of data structure has a bearing effect on the performance of the algorithm. Also give the circumstances where: **04**

(i) It is advantageous to use array instead of linked list

(ii) It is advantageous to use linked list instead of array.

Q.2 **(a)** Define Time Complexity and Space Complexity. Why we are generally concerned with Time Complexities than Space Complexities (Requirements)? What do you mean by Polynomial time complexity and Logarithmic time complexity? Which one is higher? What is a major contributor for inefficiency of a loop? **07**

(b) State Principle of *optimality*. Principle of *optimality* does not hold for all problems whose solutions can be viewed as a sequence of decisions. Find out such problem and show that it fails. **07**

OR

(b) Prove that *Greedy Algorithms* does not always give optimal solution. What are the general characteristics of *Greedy Algorithms*? Also compare *Greedy Algorithms* with *Dynamic Programming* and *Divide and Conquer* methods to find out major difference between them. **07**

Q.3 **(a)** What are the basic ideas behind string matching? Which are the two popular algorithms for string matching? Explain and analyze any one in brief. **06**

(b) Define the class **P** and class **NP**. Is there any NP-Hard problem, which is also NP? If yes, give an example, if no, why? **04**

(c) Derive the recurrence relation for quick sort in best case and find the complexity. **04**

OR

Q.3 **(a)** What is the main advantage of back-tracking? How does it achieve this? **02**

(b) Solve the following recurrence equations. **06**

1. $T(n) = T(n-1) +2T(n-2) – 2T(n-3)$
2. $T(n) = 2T(N/2) + n \log n$

	(c)	Justify the general statement that "if a problem can be split using Divide and Conquer strategy in almost equal portions at each stage, then it is a good candidate for recursive implementation, but if it cannot be easily be so divided in equal portions, then it better be implemented iteratively". Explain with an example.	**06**
Q.4	**(a)**	Prove/Give counter example: A graph with *n* nodes and more than *n-1* edges must contain cycle.	**04**
	(b)	Given a finite set of distinct coin types, say 50,20,10,5,2,1, and an integer constant C. Each type is available in unlimited quantity. Write an algorithm to find the exact change with minimum number of coins? Which strategy will you use?	**06**
	(c)	Depth First Search (DFS) algorithm can be used to find the connected components of an undirected graph. Justify the statement with reason.	**04**
		OR	
Q.4	**(a)**	Give and explain *Kruskal's Algorithm* for Minimum Spanning Tree and Compare it with *Prim's algorithm* with an example.	**07**
	(b)	Develop an algorithm and program (recursive function) to calculate the GCD of two integers using Top-Down Design. Analyze the algorithm.	**07**
Q.5	**(a)**	Write an efficient algorithm to find exponentiation $X= a^n$, where a and n are integers. Give the detailed analysis and worst case behavior for the same.	**07**
	(b)	Write an algorithm to find binomial coefficients using recursion. Analyze it.	**07**
		OR	
Q.5	**(a)**	Write a recursive algorithm to find Fibonacci numbers. Write recurrence equation for it. Solve the recurrence equation and find out the complexity.	**05**
	(b)	Give the algorithm with example to solve 0/1 Knapsack Problem using Dynamic Programming.	**05**
	(c)	Give the algorithm to find the best way to multiply n matrices. Analyze the algorithm and give the timing analysis.	**04**